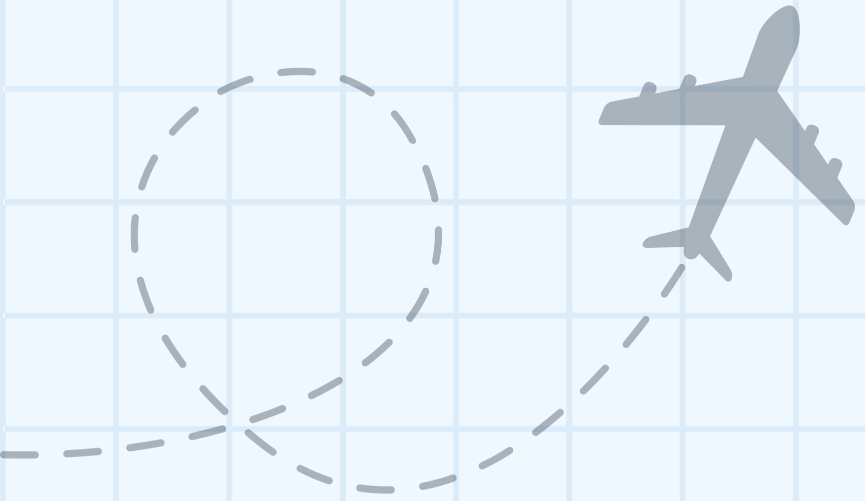
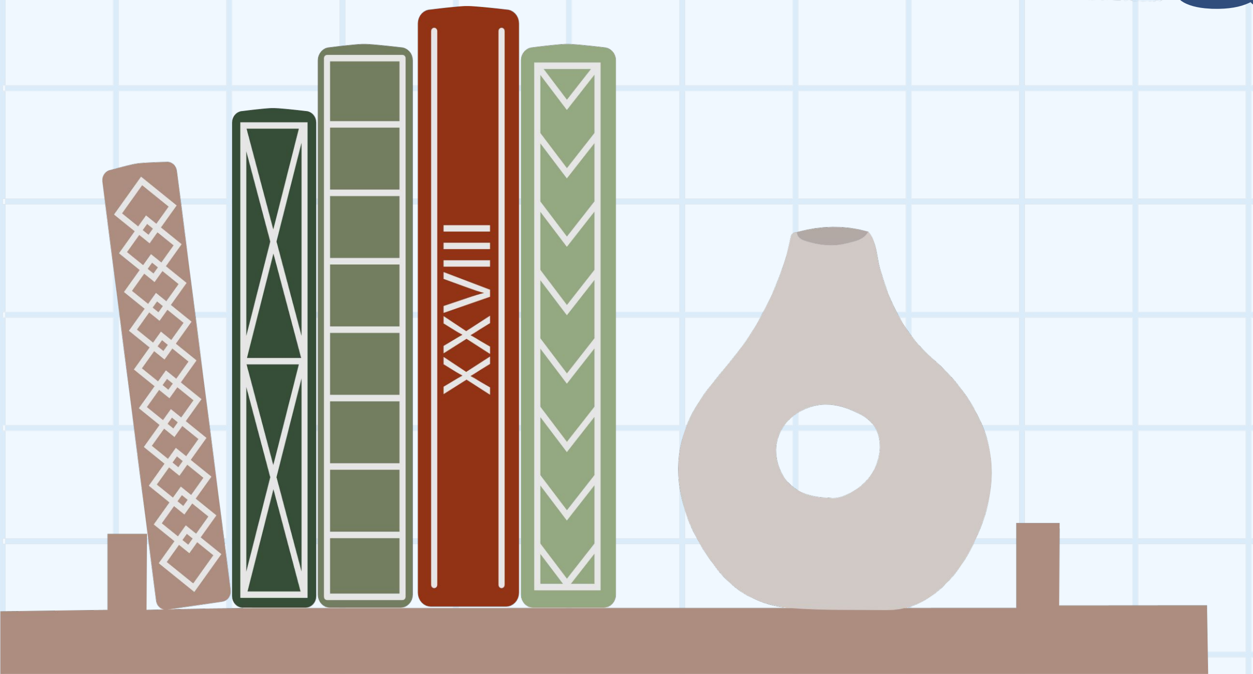


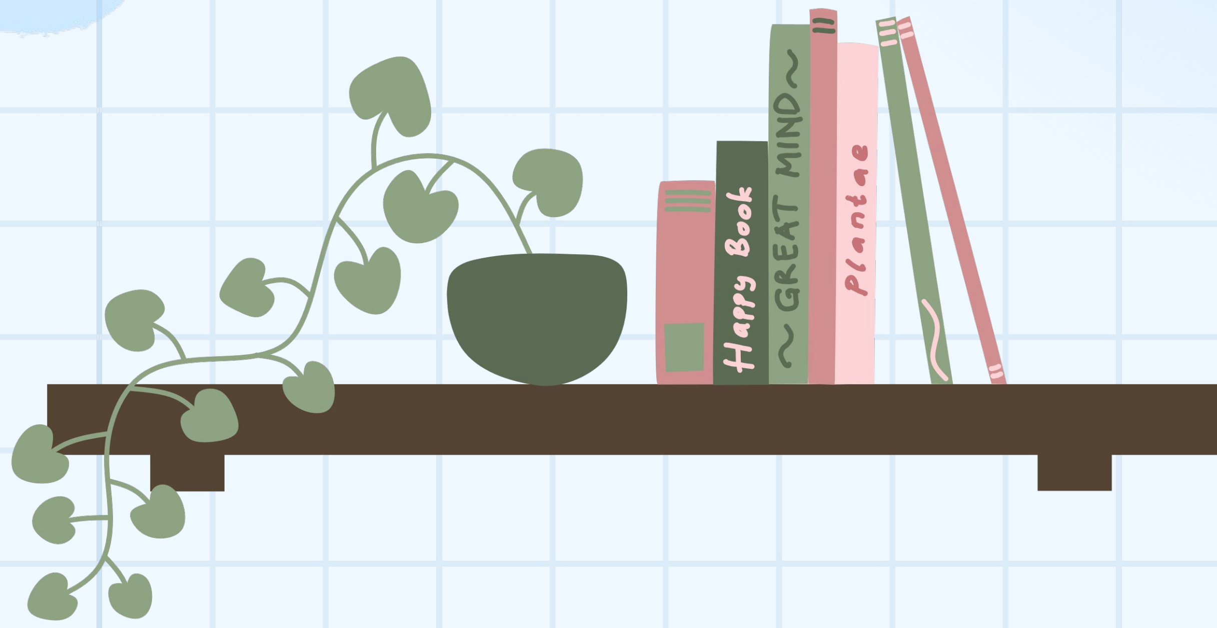
BS./BSC.

Applied AI and Data Science

Algorithmic Thinking & its Applications



Powered by
Futurense



Problem: Summing the Elements of a List

```
def sum(the_list):
    """Returns: the sum of all elements in
    the_list Precondition: the_list is a list
    of all numbers (either floats or ints)"""
```

Approach: Summing the Elements of a List

```
def sum(the_list):
    """Returns: the sum of all elements in
    the_list Precondition: the_list is a list
    of all numbers (either floats or ints)"""
    # Create a variable to hold result (start
    at 0) # Add each list element to
    variable

    # Return the variable
```

How will we do this?

1st Attempt: Summing the Elements of a List

```
def sum(the_list):
    """Returns: the sum of all elements in the_list
    Precondition: the_list is a list of all numbers
    (either floats or ints)"""
    result = 0
    result = result + the_list[0]
    result = result + the_list[1]
    …
    return result
```

We have a problem

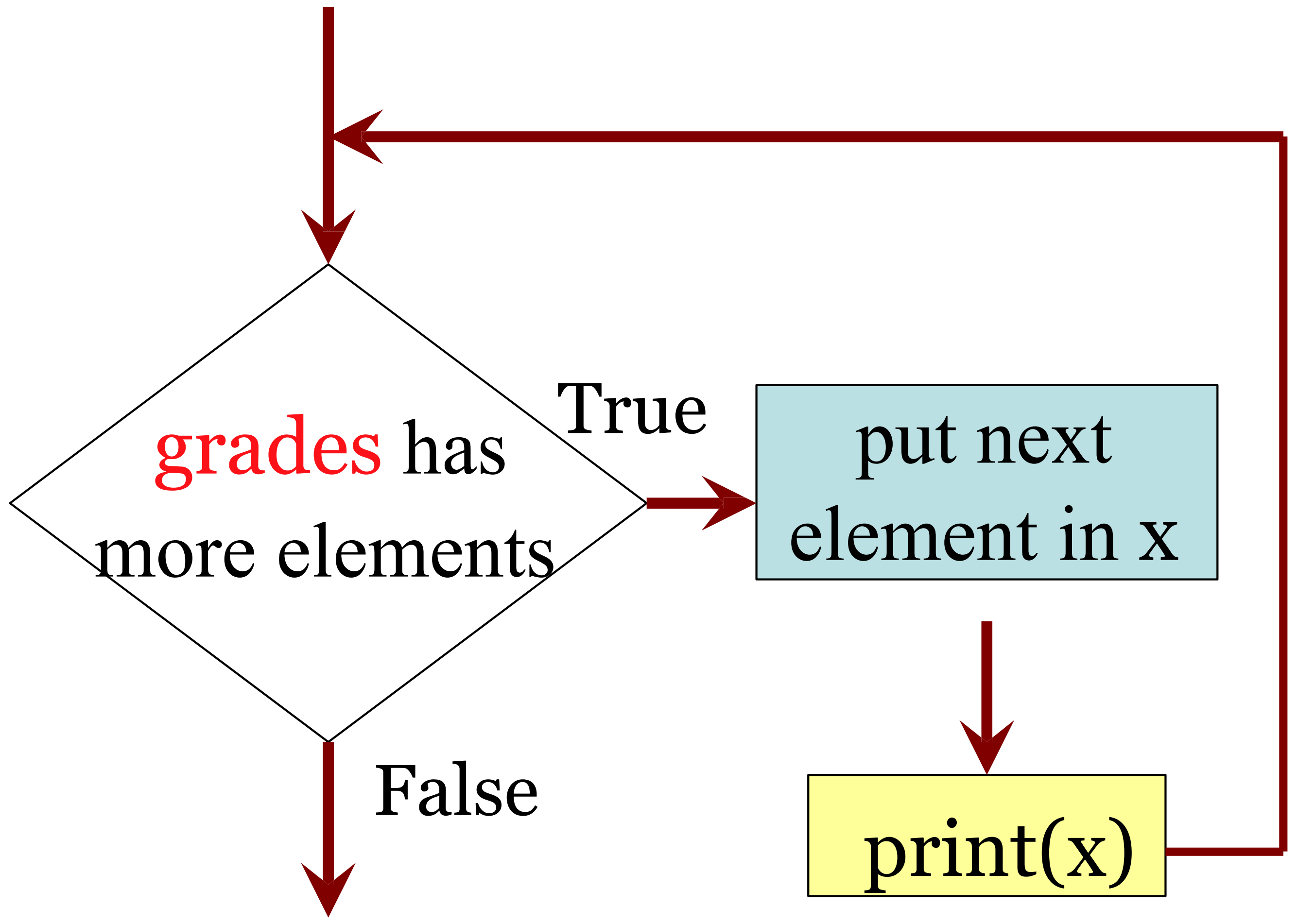
Working with Sequences

- Sequences are potentially **unbounded**
  - Number of elements is not fixed
  - Functions must handle sequences of different lengths
  - **Example**: sum([1,2,3]) vs. sum([4,5,6,7,8,9,10])
- Cannot process with **fixed** number of lines
  - Each line of code can handle at most one element
  - What if there are millions of elements?
- We need a new approach

For Loops: Processing Sequences

```
for x in grades:
    print(x)
```

- **loop sequence:** grades
- **loop variable**: x
- **body**: print(x)



To execute the for-loop:

1. Check if there is a “next” element of **loop sequence**
2. If so:
   - *assign* next sequence element to **loop variable**
   - Execute all of **the body**
   - Go back to Line 1
3. If not, terminate execution

Solution: Summing the Elements of a List

```
def sum(the_list):
    """Returns: the sum of all elements in the_list
    Precondition: the_list is a list of all numbers
    (either floats or
    ints)"""
    result = 0
    for x in the_list:
        result = result + x

    return
    result
```

Accumulator variable

- **loop sequence:** the_list
- **loop variable**: x
- **body**: result=result+x

What gets printed? (Q1)

```
a = 0
for b in
   [1]: a =
   a + 1

print(
a)
```

```
a = 0
for b in [1,
   2]: a = a
   + 1

print(
a)
```

```
a = 0
for b in [1, 2,
   3]: a = a + 1

print(
a)
```

```
a = 0
for b in [1, 2,
   3]: a = b

print(
a)
```

What gets printed? (A1)

```
a = 0
for b in [1]: a = a + 1
print(a)
```

```
1
```

```
a = 0
for b in [1, 2]: a = a + 1
print(a)
```

```
2
```

```
a = 0
for b in [1, 2, 3]: a = a + 1
print(a)
```

```
3
```

```
a = 0
for b in [1, 2, 3]: a = b
print(a)
```

```
3
```

What gets printed? (Q2)

```
a = 0
for b in [1, 2,
    3]: a = a +
    b
print(
a)
```

```
a = 0
b = [1, 2, 3]
for c in b: a
    = a + c


print(
a)
```

```
a = 0
b = [1, 2,
3]
for c in b:
    a = a +
    c
print(
b)
```

What gets printed? (A2)

```
a = 0
for b in [1, 2,
    3]: a = a +
    b
print(
a)
```

```
6
```

```
a = 0
b = [1, 2, 3]
for c in b:
    a = a + c

print(
a)
```

```
6
```

```
a = 0
b = [1, 2, 3]
for c in b:
    a = a + c

print(
b)
```

```
[1, 2, 3]
```

For Loops and Conditionals

```
def num_ints(the_list):
    """Returns: the number of ints in the_list
    Precondition: the_list is a list of any mix of
    types"""                        # Create variable to hold result
    result = 0                        # for each element in the
    for x in the_list:                list...
        if type(x) == int:              # check if it is an int
            result = result+1           # add 1 if it is
    return                            # Return the variable
    result
```

For Loop with labels

```
def num_ints(the_list):
    """Returns: the number of ints in the_list
    Precondition: the_list is a list of any mix of types"""
    result = 0
    for x in the_list:
        if type(x) == int:
            result = result+1
    return result
```

Accumulator variable

Loop sequence

Loop variable

Body

What if we aren’t dealing with a list?

So far we’ve been building for-loops around elements of a list.

What if we just want to do something some number of times?

range to the rescue!

range: a handy counting function!

range(x)

returns 0,1,…,x-1

```
>>>first_six = list(range(6))
>>> print(first_six)
[0, 1, 2, 3, 4, 5]
```

range(a,b)

returns a,…,b-1

```
>>> second_six = list(range(6,13))
>>> print(second_six) [6,
7, 8, 9, 10, 11, 12]
```

Important: **range does not return a list**

 need to convert ranges’ return value into a list

range in a for-loop, v1

```
for num in list(range(10)):
    line = "The ants go marching "+str(num)+" by "+str(num) for y in list(range(2)):
        print(line+" Hurrah! Hurrah!")
  print(line+", blah blah something that rhymes with "+str(num))
  print("And they all go marching down into the ground") print("  to get out of the rain\n")
```

Anything weird here?

(Kids don’t usually count from 0….)

range in a for-loop, v2

```
for num in l~~ist(range(10)):~~ list(range(1,11)):
    line = "The ants go marching "+str(num)+" by "+str(num) for y in list(range(2)):
        print(line+" Hurrah! Hurrah!")
    print(line+", blah blah something that rhymes with "+str(num))
    print("And they all go marching down into the ground") print("  to get out of the rai
```

Ahh, much better….

Roses

```
# at our 1 year anniversary my partner gave me a rose
# and promised to give me 1 more rose each year thereafter # how many
roses will that be?!


met_year = 2003
n_years = 75
total_roses = 0
for n_years in list(range(1, n_years+1)):
    print(str(met_year+n_years)+”: "+str(n_years)+" roses") total_roses
    = total_roses + n_years
print("After "+str(n_years)+" years: "+str(total_roses)+" roses!")
```

What gets printed? (Q3)

```
a = 0
for b in range(0,
   1): a = a + 1


print(a)
```

```
a = 0
for b in range(0,
   4): a = a + 1


print(
a)
```

What gets printed? (A3)

```
a = 0
for b in range(0,
   1): a = a + 1


print(a)
```

```
1
```

```
a = 0
for b in range(0,
   4): a = a + 1


print(
a)
```

```
4
```

Modifying the Contents of a List

```
def add_one(the_list):
    """Adds 1 to every element in a list of all numbers
    (either floats or ints)"""
    size = len(the_list)
    for k in list(range(size)):
        the_list[k] = the_list[k]+1
```

```
grades = [8,9,10,5,9,10]
print("Initial grades are: "+str(grades))
add_one(grades)
print("Inflated grades are: "+str(grades))
```

Common For-Loop Mistakes

Never modify:

(1) the loop sequence (or the list of indices) as you walk through it

(2) the loop variable

See examples on following slides.

For-Loop Mistake #1 (Q)

Modifying the loop sequence as you walk through it.

```
b = [1, 2, 3]
for a in b:
    b.append(a)


print b
```

A: never prints b

B: [1, 2, 3, 1, 2, 3]

C: [1, 2, 3]

D: I do not know

For-Loop Mistake #1 (A)

Modifying the loop sequence as you walk through it.

```
b = [1, 2, 3]
for a in b:
    b.append(a)

    INFINITE LOOP!

print b
```

A: never prints b **CORRECT***

B: [1, 2, 3, 1, 2, 3]

C: [1, 2, 3]

D: I do not know

*** Runs out of memory eventually, then probably throws an error.**

For-Loop Mistake #2 (Q)

Modifying the loop variable (here: x).

```
def add_one(the_list):
    """Adds 1 to every element in the list Precondition:
    the_list is a list of all numbers (either floats or ints)"""

    for x in the_list:
        x = x+1


a = [5, 4, 7]
add_one(a)
print(a)
```

What gets printed?

A: [5, 4, 7]
B: [5, 4, 7, 5, 4, 7]
C: [6, 5, 8]
D: **Error**
E: I don’t know

For-Loop Mistake #2 (A)

Modifying the loop variable (here: x).

Actually it does not do this!

```
def add_one(the_list):
    """Adds 1 to every element in the list
    Precondition: the_list is a list of all
    numbers (either floats or ints)"""
    for x in the_list:
        x = x+1

a = [5, 4, 7]
add_one(a)
print(a)
```

What gets printed?

A: [5, 4, 7] **CORRECT**

B: [5, 4, 7, 5, 4, 7]

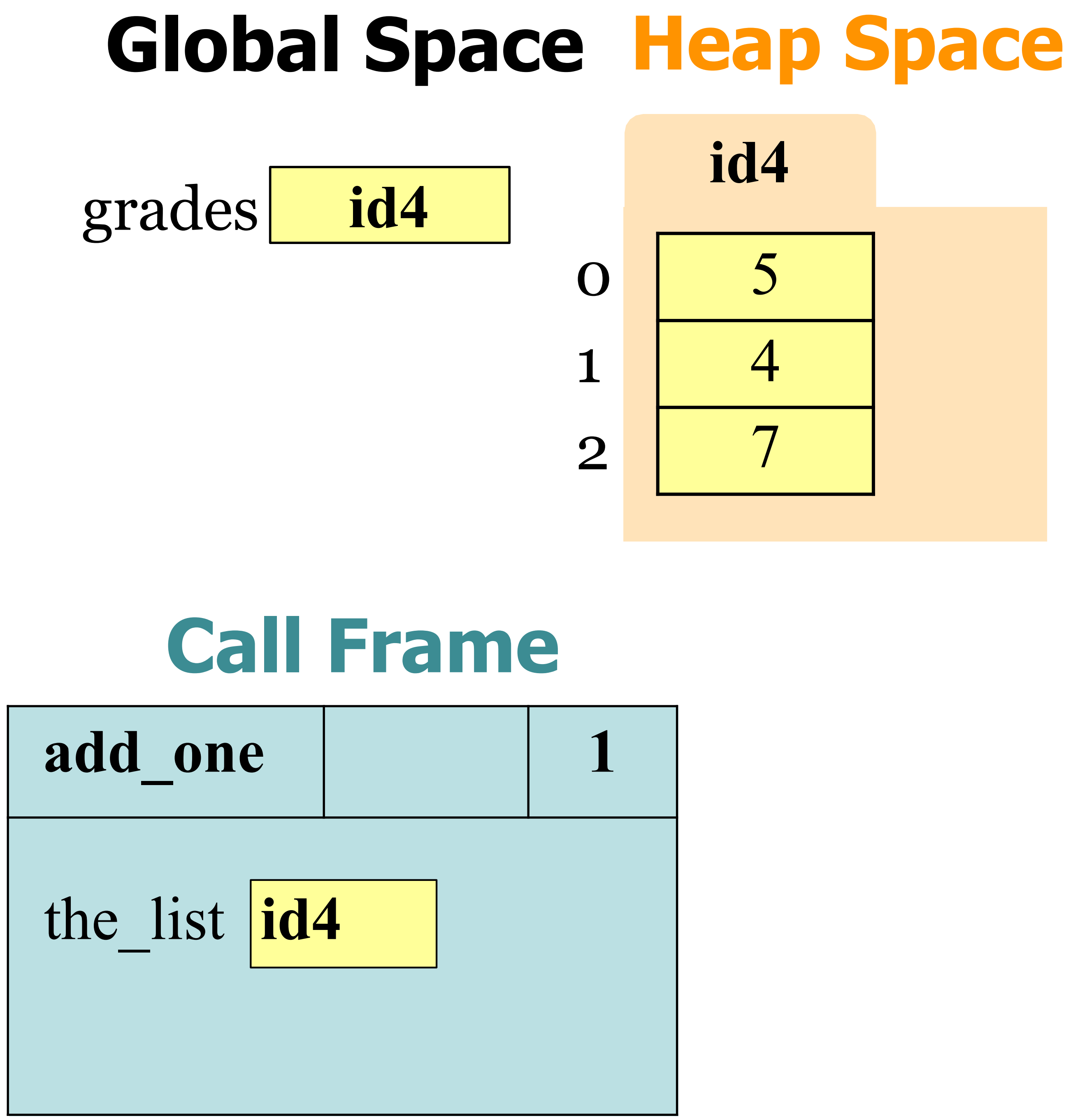
C: [6, 5, 8]

D: **Error**

E: I don't know

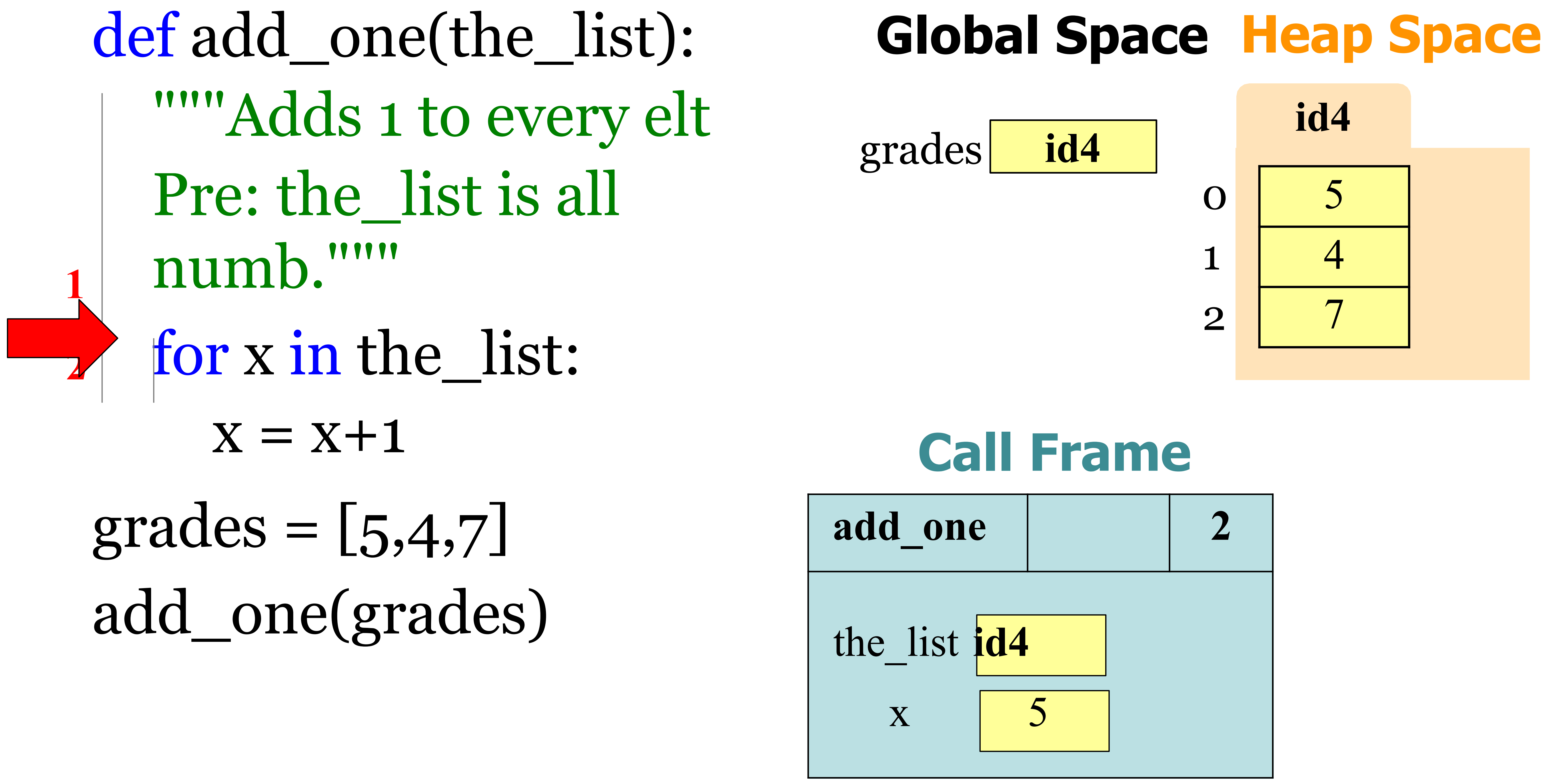
Modifying the Loop Variable (1)

```
def add_one(the_list):
        """Adds 1 to every elt
        Pre: the_list is all
1       numb."""
2       for x in the_list:
            x = x+1

grades = [5,4,7]
add_one(grades)
```



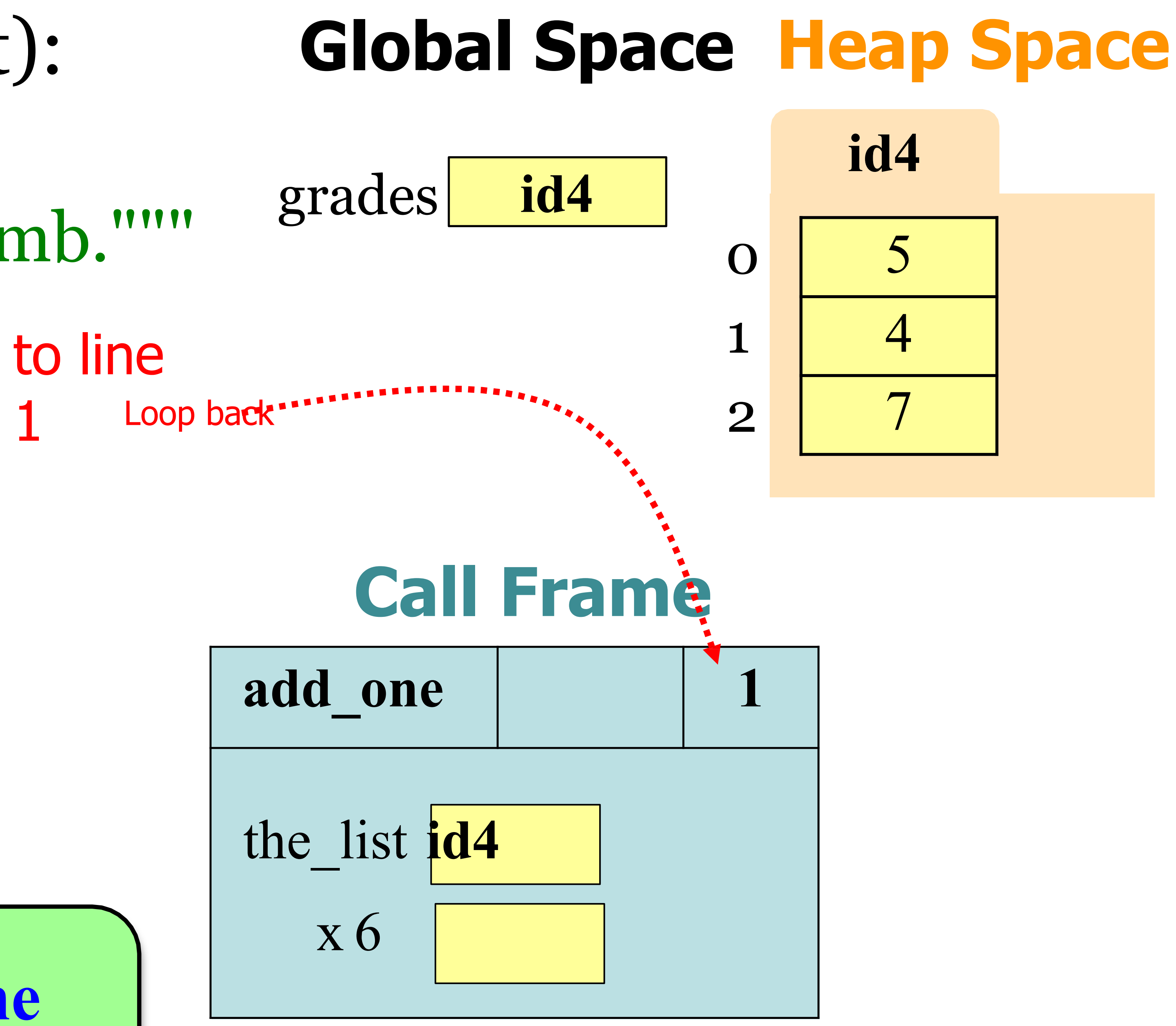
Modifying the Loop Variable (2)

```
def add_one(the_list):
    """Adds 1 to every elt
    Pre: the_list is all
    numb."""
    for x in the_list:
        x = x+1

grades = [5,4,7]
add_one(grades)
```



Modifying the Loop Variable (3)

```
    def add_one(the_list):
        """Adds 1 to every elt
        Pre: the_list is all numb."""
1       for x in the_list:
2
            x = x+1
    grades = [5,4,7]
    add_one(grades)
```

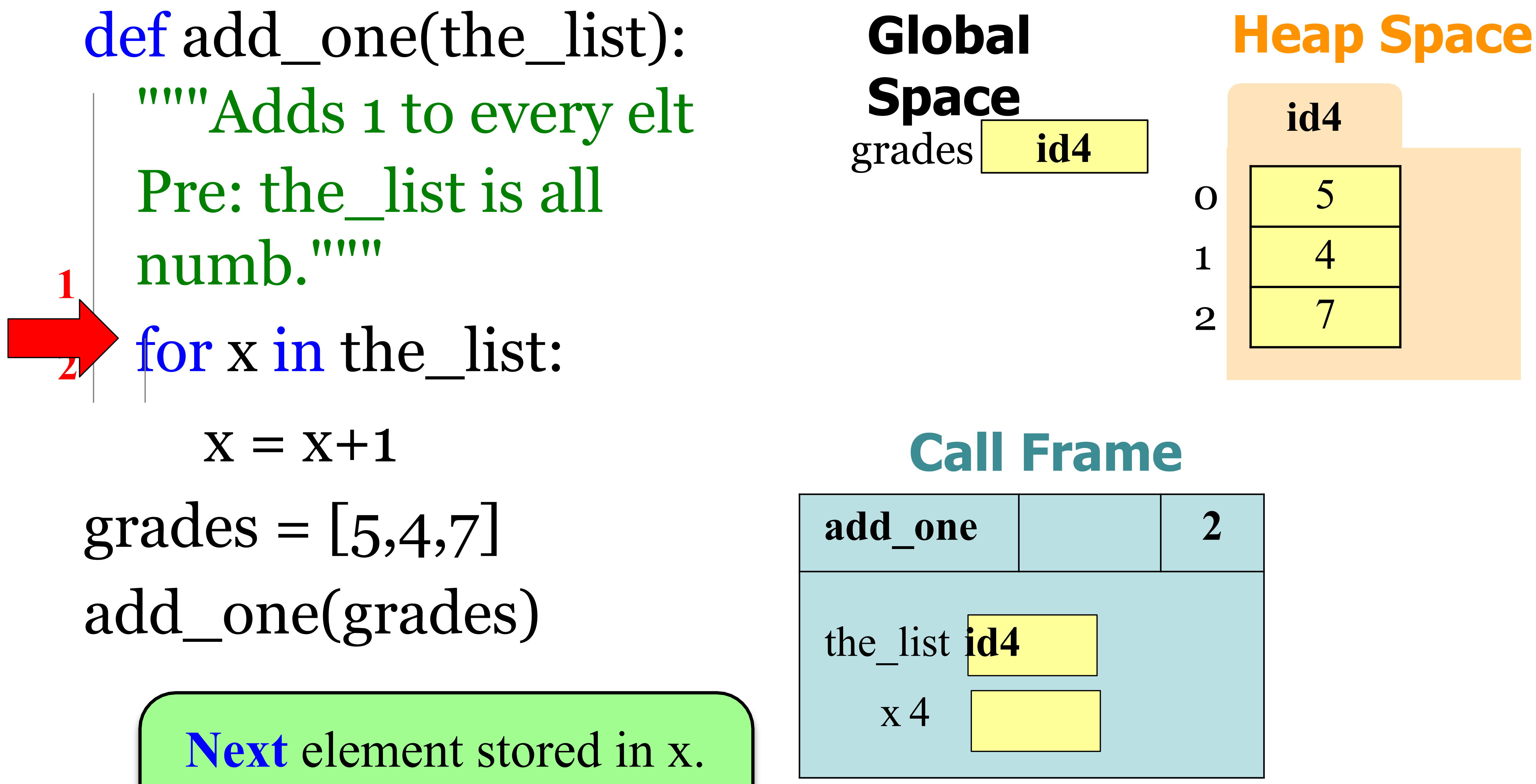


Increments x in **frame**

Does not affect folder

Modifying the Loop Variable (4)

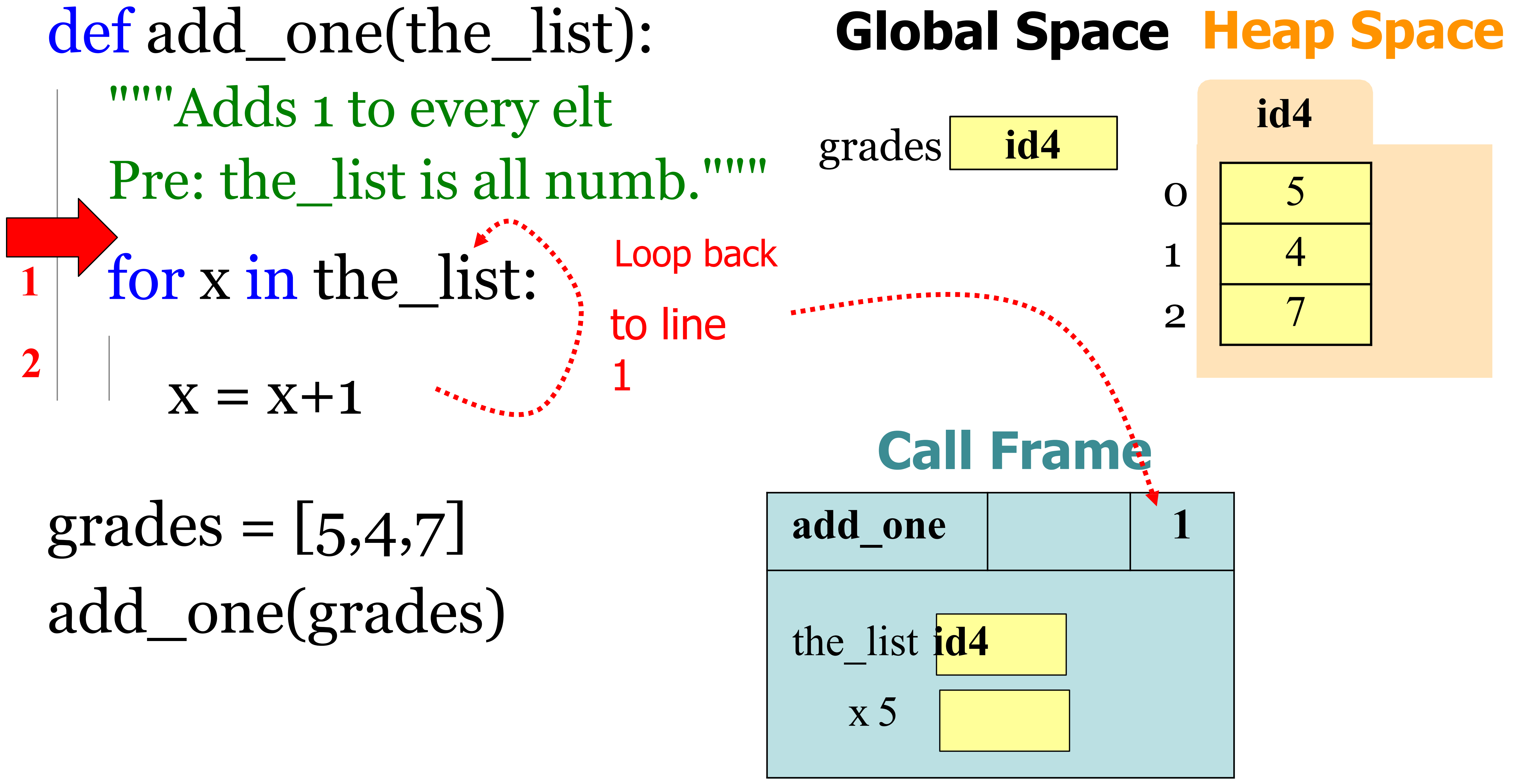
```
def add_one(the_list):
    """Adds 1 to every elt
    Pre: the_list is all
    numb."""
    for x in the_list:
        x = x+1
grades = [5,4,7]
add_one(grades)
```



Next element stored in x.
Previous calculation lost.

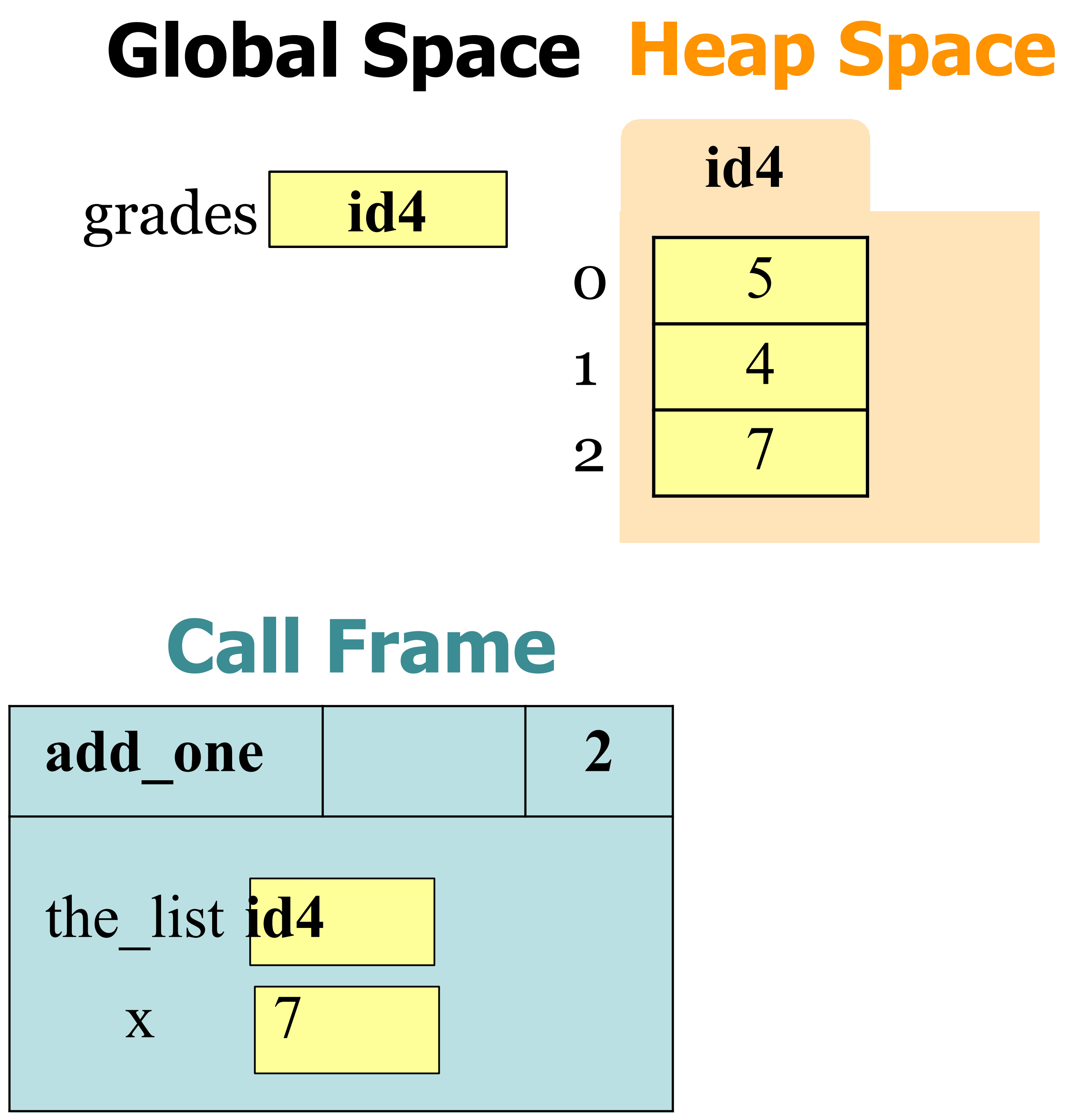
Modifying the Loop Variable (5)

```
def add_one(the_list):
    """Adds 1 to every elt
    Pre: the_list is all numb."""
1   for x in the_list:
2       x = x+1

grades = [5,4,7]
add_one(grades)
```



Modifying the Loop Variable (6)

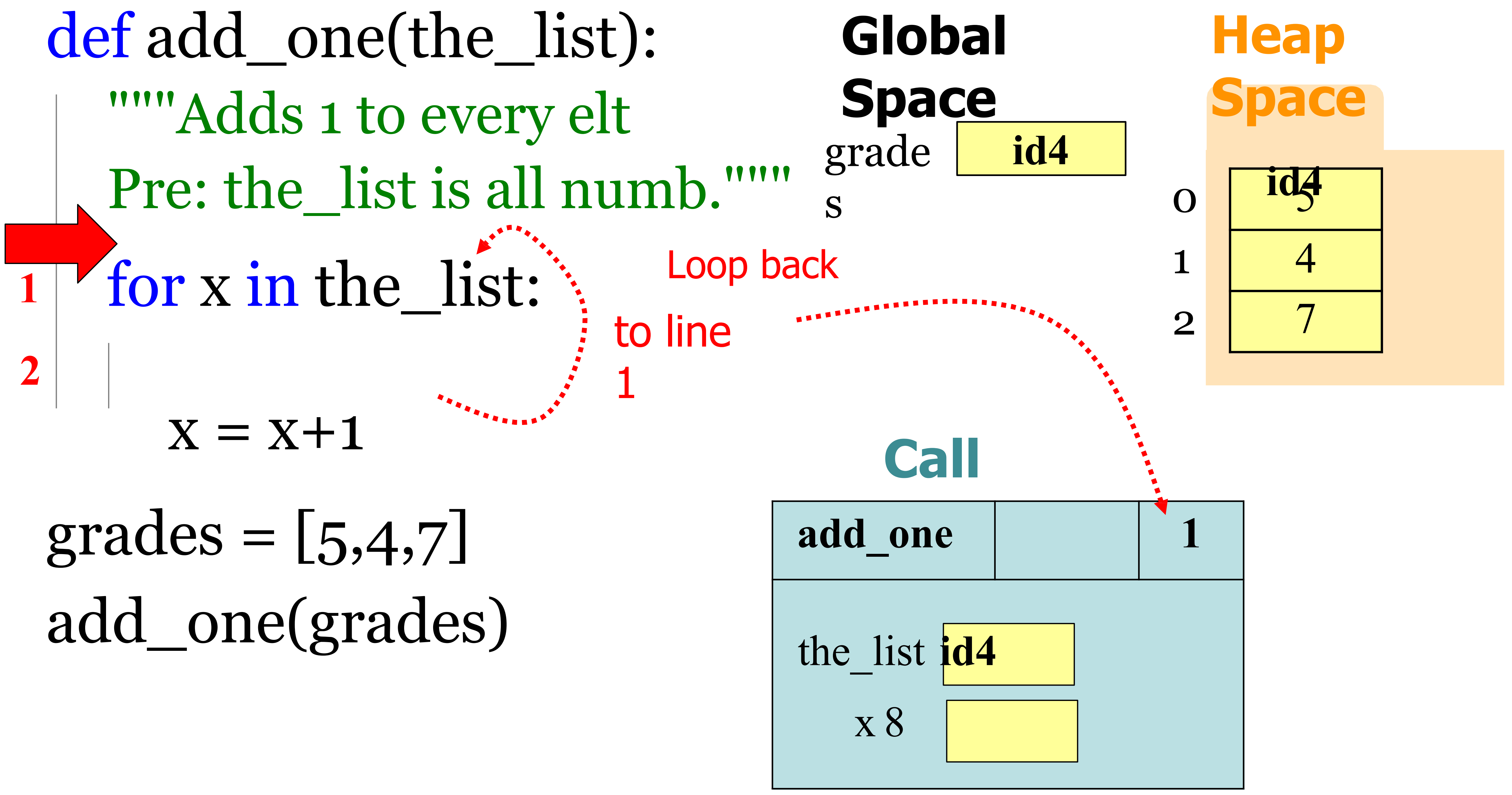
```
def add_one(the_list):
    """Adds 1 to every elt
    Pre: the_list is all
    numb."""
    for x in the_list:
        x = x+1

grades = [5,4,7]
add_one(grades)
```



Next element stored in x.
Previous calculation lost.

Modifying the Loop Variable (7)

```
def add_one(the_list):
    """Adds 1 to every elt
    Pre: the_list is all numb."""
1   for x in the_list:
2       x = x+1

grades = [5,4,7]
add_one(grades)
```

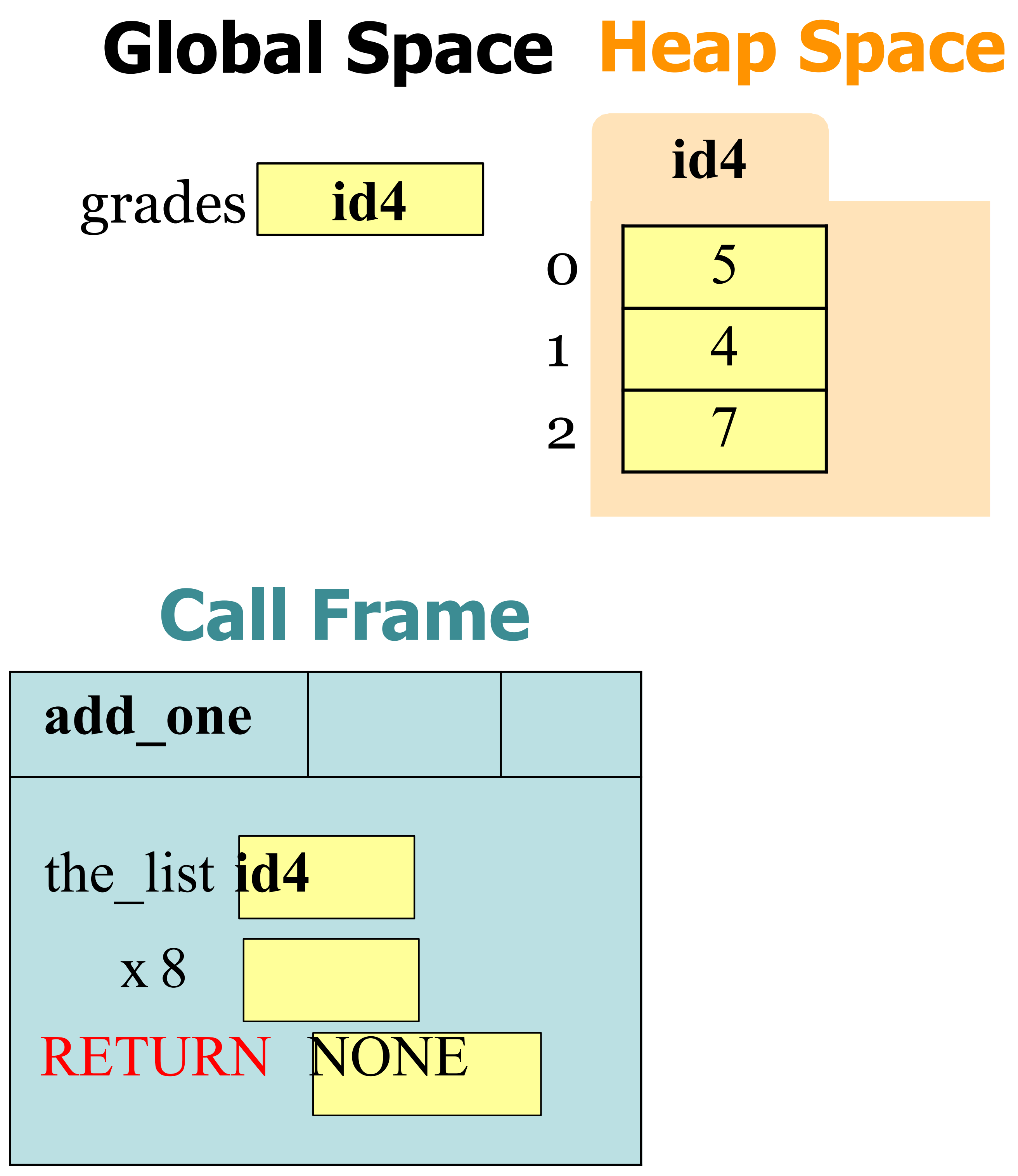


Modifying the Loop Variable (8)

```
  def add_one(the_list):
      """Adds 1 to every elt
      Pre: the_list is all
1     numb."""
2     for x in the_list:
          x = x+1
  grades = [5,4,7]
  add_one(grades)
```

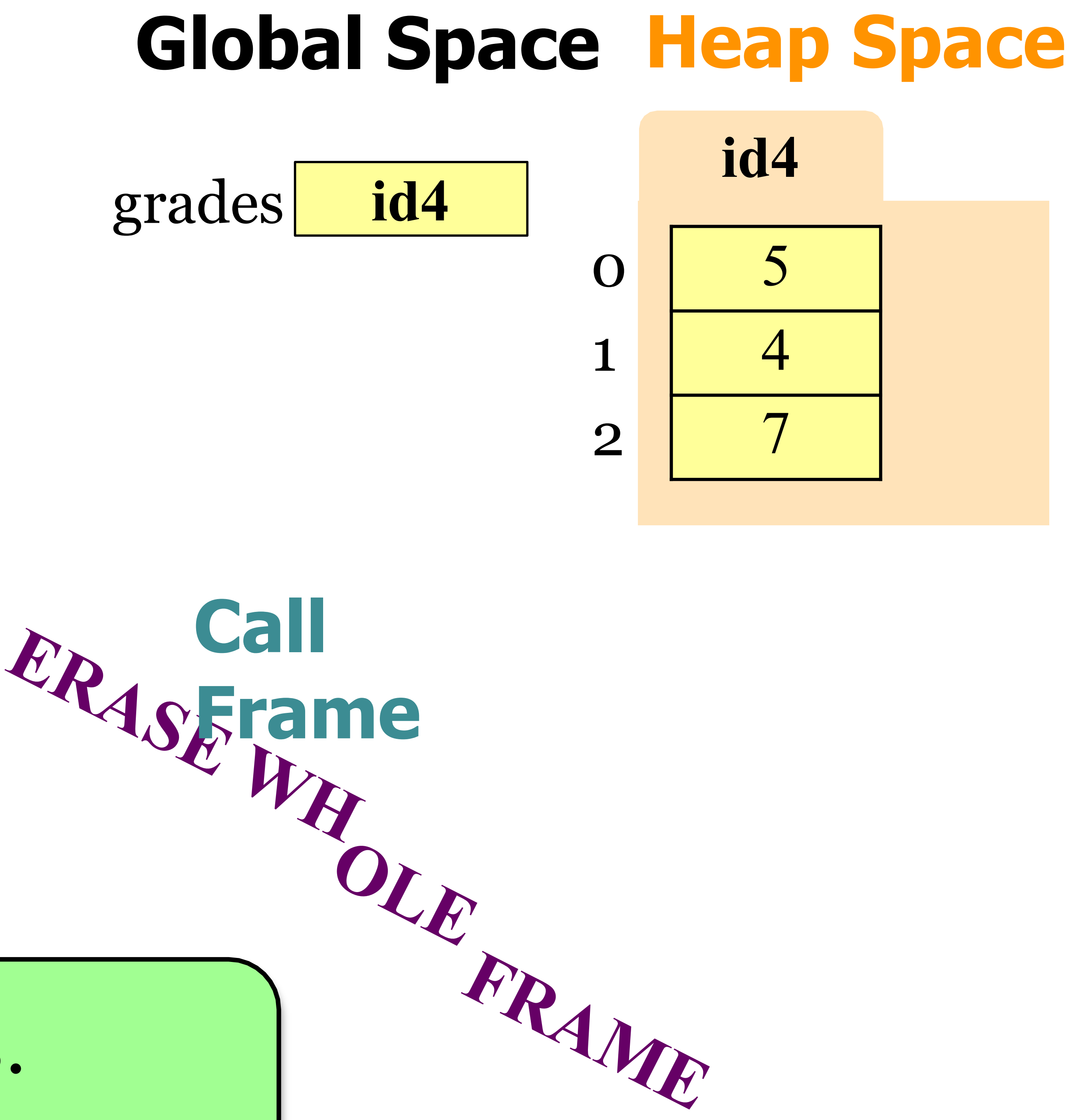
Loop is **completed.**

Nothing new put in x.



Modifying the Loop Variable (9)

```
  def add_one(the_list):
      """Adds 1 to every elt
      Pre: the_list is all
1     numb."""
2     for x in the_list:
          x = x+1

  grades = [5,4,7]
  add_one(grades)
```



No lasting changes.

What did we accomplish? ☹

Thank you